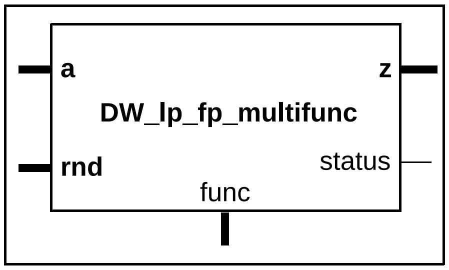
DW_lp_fp_multifunc

Low Power Floating-Point Multi-Function Unit

Version, STAR, and myDesignWare Subscriptions: IP Directory

Features and Benefits

Revision History



- The precision format can be set for single precision or a user-defined custom floating-point format.
- Hardware for denormal numbers of IEEE 754 standard is selectively provided.
- It implements any combination of the following functions:

$$\frac{1}{x}, \frac{1}{\sqrt{x}}, \sqrt{x}, \sin(\pi x) \text{ or } \sin(x), \cos(\pi x) \text{ or } \cos(x), \log_2(x), \text{and } 2^x$$

- One function is computed at a time.
- All functions produce monotonic results (depends on input range--see Table 1-6 on page 4).
- Shared polynomial approximation unit provides a solution with reduced area.

Note You must use VCS to simulate the DW_lp_fp_multifunc component. Non-VCS simulators are not supported.

Description

DW_lp_fp_multifunc is a floating-point multi-function unit that implements any combination of seven functions: reciprocal, square root, reciprocal square root, sine, cosine, base-2 logarithm and base-2 exponential. The particular set of functions to be implemented is selected with the *func_select* parameter as a one-hot value. At any given time the unit computes one function in the set defined by input `func`.

The input `rnd` takes effect only when the reciprocal function is invoked with the input `func` = 1 and the parameter *faithful_round* = 0. The parameter *pi_multiple* is valid only when sine or cosine function is selected. Both input `a` and output `z` have the floating-point format, and the output `status` is an 8-bit optional status field, which are described in the *Datapath Floating-Point Overview*.

Table 1-1 Pin Descriptions

Pin Name	Width	Direction	Function
a	(*sig_width* + *exp_width* + 1) bits	Input	Input data
func	16 bit	Input	Function selection
rnd	3 bit	Input	Rounding mode for reciprocal function (only valid when `func` = 1)
z	(*sig_width* + *exp_width* + 1) bits	Output	Output data
status	8 bits	Output	■ Status flags for the result; for details, see STATUS Flags in the *Datapath Floating-Point Overview* ■ `status`[7]: Indicates divide-by-zero operation of reciprocal, reciprocal square root, and logarithm functions.

Table 1-2 Parameter Descriptions

Parameter	Values	Description
sig_width	2 to 23 Default: 23	Word length of fraction field of floating point numbers a and z
exp_width	3 to 8 Default: 8	Word length of biased exponent of floating point numbers a and z
ieee_compliance	0 or 1 Default: 0	When 1, the generated architecture includes the use of denormals and NaNs
func_select	1 to 127 Default: 127	Determines the functions to be implemented among the seven functions (one bit for each function)
faithful_round	0 or 1 Default: 1	Selects the faithful rounding mode ■ 0 = Supports all rounding modes described in the *Datapath Floating-Point Overview*) ■ 1 = Results have maximum of 1 *ulp* error
pi_multiple	0 or 1 Default: 1	Input value (Angle) is multiplied by π; this parameter is only valid when sin or cos functions are selected ■ 0: `z` = sin(`a`) or cos(`a`) ■ 1: `z` = sin(π`a`) or cos(π`a`)

Table 1-3 Synthesis Implementations

Implementation Name	Implementation	License Feature Required
rtl	Low Power Synthesis model	■ DesignWare (P-2019.03 and later) ■ DesignWare-LP[a] (before P-2019.03)

a. For versions before P-2019.03, you must enable minPower as follows:
```
set_synthetic_library {dw_foundation.sldb dw_minpower.sldb}
```

Table 1-4 Simulation Models

Model	Function
dw/sim_ver/DW_lp_fp_multifunc.v	Verilog simulation model[a] source code Note that you must use VCS to simulate the DW_lp_fp_multifunc component. Non-VCS simulators are not supported.

a. To use this simulation model, the '+v2k' options needs to be used on the VCS command line.

Parameter func_select and Input func

The 7-bit parameter *func_select* is used for selecting the set of functions to be implemented. The value of *func_select* has the range from 1 to 127, and each bit corresponds to a function, as show in Table 1-5.

Table 1-5 func_select Parameter Implementation (design) and func Function Selection (use)

func_select Bit Weight	Function Description	func 16-bit select value
2^0	reciprocal, 1/a	0000_0000_0000_0001
2^1	square root of a	0000_0000_0000_0010
2^2	reciprocal square root of a	0000_0000_0000_0100
2^3	sine, $\sin(\pi a)$	0000_0000_0000_1000
2^4	cosine, $\cos(\pi a)$	0000_0000_0001_0000
2^5	base-2 logarithm, $\log_2 a$	0000_0000_0010_0000
2^6	base-2 exponential, 2^a	0000_0000_0100_0000
	(2^7 - 2^{15}) reserved for future support	reserved for future support

For example, if only a reciprocal function is required, *func_select* needs to be 1 (16'h0001). If reciprocal, reciprocal square root and base-2 logarithm functions are needed, *func_select* should be 37 (16'h0025). If all seven functions are implemented, *func_select* is 127 (16'h007f).

During operation, the input `func` specifies the single function that is to be computed. The `func` port is a 16-bit input port, and it receives a one-hot encoded value. The valid one-hot codes for `func` are defined in Table 1-5.

Error Range and Monotonicity

Reciprocal, square root, reciprocal square root and base-2 exponential functions admit maximum 1 *ulp* error on the normalized significand value. However, sine, cosine and base-2 logarithm functions are not applicable to this rule, but they provide the maximum 2^{-sig_width} error. Sine and cosine functions with *pi_multiple* = 0 have up to $3*2^{-sig\_width}$ error.

The DW_lp_fp_multifunc component produces monotonic results. The input ranges under which monotonic results are produced are shown in Table 1-6.

Table 1-6 Input Range for Monotonic Results

	1/x	sqrt(x)	1/sqrt(x)	sin(x)	cos(x)	log2(x)	exp2(x)
sig_width Input Range	All values of *sig_width*	All values of *sig_width*	All values of *sig_width*	*sig_width* <= 16	*sig_width* <= 16	All values of *sig_width*	All values of *sig_width*

Denormal Support

DW_lp_fp_multifunc provides the hardware for denormal numbers and NaNs of IEEE 754 standard. If the parameter *ieee_compliance* is turned off, denormal numbers are considered as zeros, and NaNs are considered as infinity. Otherwise, denormal numbers and NaNs become effective and additional hardware to manipulate them is integrated.

For more information about floating point, including status flag bits, and integer and floating point formats, refer to the *Datapath Floating-Point Overview*.

Verilog Description Example

If all functions besides sine and cosine are implemented (*func_select* = 103) and the output needs to produce the result of square root operation, the Verilog instantiation of such component is done as follows:

```
DW_lp_fp_multifunc #(23, 8, 0, 103) U1 (
    .a(32'h3E800000),
    .rnd(3'b000),
    .func(16'h0002),
    .z(z),
    .status(status)
);
```

Note If the above module is used to calculate the sine function (`func` = 16'h0008), DW_lp_fp_multifunc cannot generate correct function values because it does not have the hardwired logic for the sine function with the parameter *func_select* = 103.

Single-function Implementation from DW_lp_fp_multifunc

Using the *func_select* parameter to choose a single function, the DW_lp_fp_multifunc generates the netlist for the single function that has the similar functionality with single-function components; for example, DW_fp_recip, DW_fp_sqrt, DW_fp_invsqrt, DW_fp_sincos, DW_fp_log2 and DW_fp_exp2. The difference between the DW_lp_fp_multifunc and the single-function component is the value of 1 ulp, due to the different implementation of polynomial approximation, but all evaluated results keep the error bound less than 1 ulp. Performance and area of the synthesis results are different depending on the synthesis constraints, library cells, and synthesis environments. Therefore, by comparing performance and area between a single-function implementation of the DW_lp_fp_multifunc and the corresponding single-function DW component, the DW_lp_fp_multifunc component provides more choices and yields better synthesis results. The following examples generate single functions.

Floating-point reciprocal (U1)

Results are the same as DW_fp_recip when *faithful_round* = 0, or will be the same except 1 ulp value when *faithful_round* = 1.

```
DW_lp_fp_multifunc #(sig_width, exp_width, ieee_compliance, 1, faithful_round, 1) U1 (
    .a(a),
    .func(16'h0001),
    .rnd(rnd),
    .z(z),
    .status(status)
);
```

Floating-point square root (U2)

Results are the same as DW_fp_sqrt except 1 ulp value.

```
DW_lp_fp_multifunc #(sig_width, exp_width, ieee_compliance, 2) U2 (
    .a(a),
    .func(16'h0002),
    .rnd(3'h0),
    .z(z),
    .status(status)
);
```

Floating-point reciprocal square root (U3)

Results are the same as DW_fp_invsqrt except 1 ulp value.

```
DW_lp_fp_multifunc #(sig_width, exp_width, ieee_compliance, 4) U3 (
    .a(a),
    .func(16'h0004),
    .rnd(3'h0),
    .z(z),
    .status(status)
);
```

Floating-point sine and cosine (U4)

Results are the same as DW_fp_sincos except 1 ulp value. Input `func` must be one of two values, 16'h0008 for sine and 16'h0010 for cosine evaluation.

```
DW_lp_fp_multifunc #(sig_width, exp_width, ieee_compliance, 24, 0, pi_multiple) U4 (
    .a(a),
    .func(func),
    .rnd(3'h0),
    .z(z),
    .status(status)
);
```

Floating-point base-2 logarithm (U5)

Results are the same as DW_fp_log2 except 1 ulp value.

```
DW_lp_fp_multifunc #(sig_width, exp_width, ieee_compliance, 32) U5 (
    .a(a),
    .func(16'h0020),
    .rnd(3'h0),
    .z(z),
    .status(status)
);
```

Floating-point base-2 exponential (U6)

Results are the same as DW_fp_exp2 except 1 ulp value.

```
DW_lp_fp_multifunc #(sig_width, exp_width, ieee_compliance, 64) U5 (
    .a(a),
    .func(16'h0040),
    .rnd(3'h0),
    .z(z),
    .status(status)
);
```

Related Topics

- Math – Arithmetic Overview
- DesignWare Building Blocks User Guide

HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DWpackages.all;
use DWARE.DW_Foundation_comp_arith.all;

entity DW_lp_fp_multifunc_inst is
      generic (
        inst_sig_width : INTEGER := 23;
        inst_exp_width : INTEGER := 8;
        inst_ieee_compliance : INTEGER := 0;
        inst_func_select : INTEGER := 127;
        inst_faithful_round : INTEGER := 1;
        inst_pi_multiple : INTEGER := 1
        );
      port (
        inst_a : in std_logic_vector(inst_sig_width+inst_exp_width downto 0);
        inst_func : in std_logic_vector(15 downto 0);
        inst_rnd : in std_logic_vector(2 downto 0);
        z_inst : out std_logic_vector(inst_sig_width+inst_exp_width downto 0);
        status_inst : out std_logic_vector(7 downto 0)
        );
    end DW_lp_fp_multifunc_inst;


architecture inst of DW_lp_fp_multifunc_inst is

begin
    -- Instance of DW_lp_fp_multifunc
    U1 : DW_lp_fp_multifunc
    generic map (
          sig_width => inst_sig_width,
          exp_width => inst_exp_width,
          ieee_compliance => inst_ieee_compliance,
          func_select => inst_func_select,
          faithful_round => inst_faithful_round,
          pi_multiple => inst_pi_multiple
          )
    port map (
          a => inst_a,
          func => inst_func,
          rnd => inst_rnd,
          z => z_inst,
          status => status_inst
          );

end inst;
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_lp_fp_multifunc_inst( inst_a, inst_func, inst_rnd, z_inst, status_inst );

parameter inst_sig_width = 23;
parameter inst_exp_width = 8;
parameter inst_ieee_compliance = 0;
parameter inst_func_select = 127;
parameter inst_faithful_round = 1;
parameter inst_pi_multiple = 1;


input [inst_sig_width+inst_exp_width : 0] inst_a;
input [15 : 0] inst_func;
input [2 : 0] inst_rnd;
output [inst_sig_width+inst_exp_width : 0] z_inst;
output [7 : 0] status_inst;

    // Instance of DW_lp_fp_multifunc
    DW_lp_fp_multifunc #(inst_sig_width, inst_exp_width, inst_ieee_compliance,
inst_func_select, inst_faithful_round, inst_pi_multiple) U1 (
                .a(inst_a),
                .func(inst_func),
                .rnd(inst_rnd),
                .z(z_inst),
                .status(status_inst) );

endmodule
```

Revision History

For notes about this release, see the *DesignWare Building Block IP Release Notes*.

For lists of both known and fixed issues for this component, refer to the STAR report.

For a version of this datasheet with visible change bars, click here.

Date	**Release**	**Updates**
January 2020	DWBB_201912.1	■ Corrected port names to be lowercase in Table 1-1 on page 2, examples, and format presentations
March 2019	DWBB_201903.0	■ Clarified some information about minPower in Table 1-3 on page 3
July 2018	DWBB_201806.1	■ For STAR 9001366625, added a note about simulator support on page 1 and in Table 1-4 on page 3 ■ Added this Revision History table and the document links on this page